HAR: Hardness Aware Reweighting for Imbalanced Datasets

Rahul Duggal
Georgia Institute of Technology
rahulduggal@gatech.edu

Scott Freitas
Georgia Institute of Technology
safreita@gatech.edu

Sunny Dhamnani
Georgia Institute of Technology
sdhamnani3@gatech.edu

Duen Horng Chau
Georgia Institute of Technology
polo@gatech.edu

Jimeng Sun
University of Illinois at Urbana Champaign
jimeng@illinois.edu

***Abstract*—Class imbalance is a significant issue that causes neural networks to underfit to the rare classes. Traditional mitigation strategies include loss reshaping and data resampling which amount to increasing the loss contribution of minority classes and decreasing the loss contributed by the majority ones. However, by treating each example within a class equally, these methods lead to undesirable scenarios where hard-to-classify examples from the majority classes are down-weighted and easy-to-classify examples from the minority classes are up-weighted. We propose the *Hardness Aware Reweighting* (`HAR`) framework, which circumvents this issue by increasing the loss contribution of hard examples from both the majority and minority classes. This is achieved by augmenting a neural network with intermediate classifier branches to enable early-exiting during training. Experimental results on large-scale datasets demonstrate that `HAR` consistently improves state-of-the-art accuracy while saving up to 20% of inference FLOPS.**

***Index Terms*—Class imbalance, neural networks, hardness, reweighting**

I. Introduction

Class imbalance is a ubiquitous phenomenon commonly observed in naturally occurring data distributions [1]–[3]. However, despite it's ubiquity, popular datasets (e.g., CIFAR-10/100 [4], ImageNet [5]) are often artificially balanced leading to a mismatch between reality and practice. Realizing this mismatch, recent research aims to account for class imbalance by evolving the traditional datasets to their *long-tailed* (LT) versions, e.g., CIFAR LT [6], ImageNet LT [7]. Here, a long tail signifies that the majority of the classes constitute only a minority of the overall data and vice versa. Our work focuses on training accurate neural networks on datasets exhibiting a long-tail class imbalance.

Overcoming a long tail imbalance is hard, since most classifiers tend to favor the majority classes that constitute the bulk of the training set [1], [8]. This is especially true for convolutional neural networks (CNNs), which are known to suffer under class imbalance [9]. An effective mitigating strategy is loss reweighting, which follows the simple rule: *increase the loss contributed by the minority classes, and decrease the loss contributed by the majority ones*. This idea is realized in popular reweighting methods, such as class reweighting based on inverse frequency [10], [11], inverse-square-root frequency [12] and the effective number of samples [6]. These methods, however, uniformly reweight all examples within each class, leading to the undesirable scenario wherein hard-to-classify examples from the majority classes are downweighted, and easy-to-classify examples from the minority classes are upweighted. This is undesirable because hard examples are known to provide stronger learning signals [13], [14] and should avoid being downweighted.

One way to avoid the above scenario is to develop a more fine-grained, instance-specific reweighting strategy. This is the motivation behind recent methods that use meta-learning [15], [16] and domain-adaptation [17]. However, these methods do not encode the key notion of example hardness, which means that hard examples are still susceptible to being downweighted, causing their learning signals to be diminished [13], [14]. On the other end, classical hardness-aware methods such as ADASYN [18] and SMOTEBoost [19] do not scale to modern CNNs [9]. This presents a need for developing a *hardness-aware*, instance-specific reweighting strategy suitable for training deep neural networks.

We propose the **H**ardness **A**ware **R**eweighting (**`HAR`**) framework that incorporates the notion of example hardness during training. `HAR` is premised around the idea of increasing the loss contribution of hard examples in both the majority and minority classes. It augments a backbone neural network with auxiliary classifier branches (illustrated in Fig. 1) which enable the backbone to learn a notion of hardness by conditionally exiting easy-to-classify examples during training. Once an example exits, it no longer incurs additional loss. A natural outcome of this process is that harder examples exit towards the end and accumulate a higher overall loss, thus realizing `HAR`'s goal.

A. Our Contributions

1. Example Hardness as Key to Unlock Generalization. We contribute the key idea that accounting for *example hardness*

978-1-6654-3902-2/21/$31.00 ©2021 IEEE

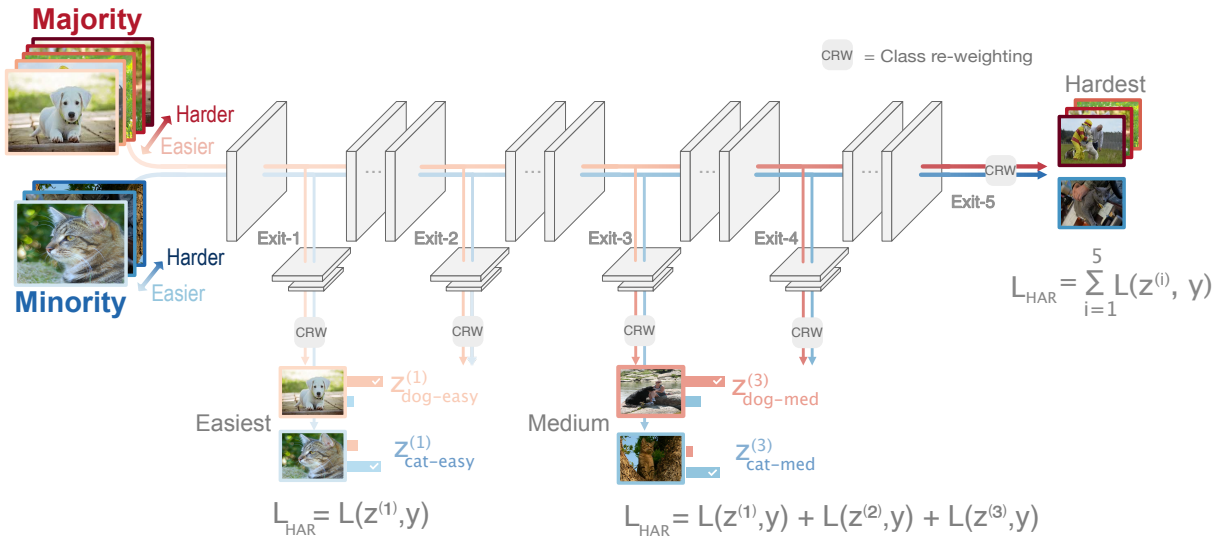


Fig. 1: **H**ardness **A**ware **R**eweighting (**HAR**) framework augments a backbone network with auxiliary classifier branches. During training, an example accumulates loss at each branch, until either (a) it is *confidently* and *correctly* classified at a branch, or (b) it reaches the end. A harder example exits later in the network and accumulate higher overall loss.

during class reweighting can improve accuracy of modern CNNs trained under long-tail class imbalance. (Sec. III-A)

2. HAR: General Framework to Endow Hardness Awareness. Our proposed HAR framework is a general approach to endow any loss function with hardness awareness and offers three benefits:

1) **State-of-the-art Accuracy**. By increasing the loss contribution of hard examples, HAR shifts the focus of learning to harder examples, leading to state-of-the-art accuracies on standard, large-scale imbalanced benchmark datasets, such as ImageNet LT [7] and iNaturalist '18 [20]. (Sec. IV-D)
2) **Compute Savings**. HAR enables *dynamic* inference wherein the inference cost of a model can be varied in realtime. The blue dots (•) in the Fig. 2 plots a HAR model's accuracies at eight different compute budgets. The curve formed envelopes all other approaches, showing that HAR enables inference FLOPS savings while still achieving state-of-the-art accuracy. (Sec. III-C)
3) **Plug-and-play Support for Existing Loss functions.** HAR can endow any loss functions with hardness awareness by using it at the auxiliary branches. We observe large accuracy improvements when using the weighted cross-entropy [6] or the weighted LDAM loss [21] at the exits. (Sec. III-B)

3. Extensive Evaluation on Large-scale Datasets. We comprehensively evaluate HAR using modern CNNs by training them on four standard imbalanced benchmarks, including the large-scale ImageNet LT [7] and iNaturalist '18 [20] datasets. Additionally, we ablate on the different modelling choices of HAR to uncover why and how it leads to a higher accuracy with compute savings. (Sec. IV-F)

Conceptually, HAR shares its motivation with the seminal work of focal loss [22] which is the first to demonstrate the benefit of enabling hardness awareness in the class imbalance

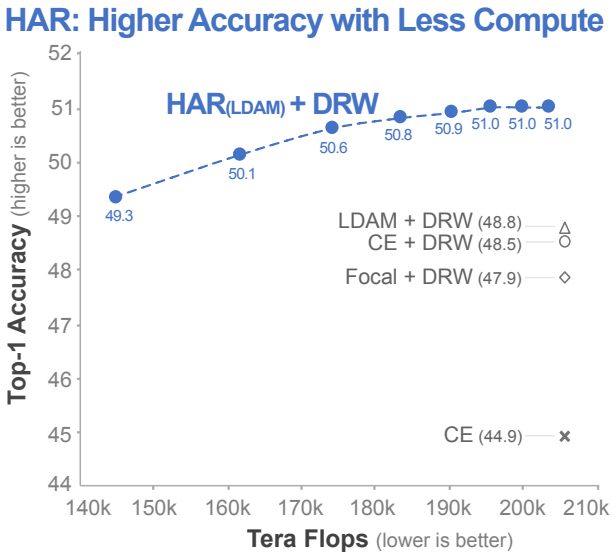


Fig. 2: The HAR framework leads to higher top-1 accuracies while saving compute, for a ResNet-50 model trained on the ImageNet LT dataset. Additionally, HAR supports *dynamic* inference that offers a favorable top-1 accuracy *vs*. efficiency trade-off under different compute budgets (shown as blue dots •). In contrast, traditional methods lead to *static* models with *fixed* compute costs during inference.

setting. There are however two major differences: (1) focal loss uses prediction confidence as a measure of hardness, while HAR uses the idea of early exiting. This couples focal loss to cross entropy, whereas HAR can work with many loss types; (2) unlike HAR, focal loss cannot save compute on easy examples, as early-exiting is not an option. Our experiments in Sec. IV show that a model trained with HAR significantly outperforms

a model trained with focal loss.

II. Related Research

We summarize related works from three relevant areas: class imbalance, hardness aware learning, and multi-branch neural networks.

A. Overcoming class imbalance

The techniques for overcoming class imbalance techniques fall into the following three categories.

Loss rebalancing: These methods reweight the loss contribution of each example such that the loss for minority classes is upweighted, while that of the majority classes is downweighted. The weighting scheme itself can be uniform across all the examples within a class [6], [23] or can be more fine-grained *i.e.*,specific to each example in consideration [15]–[17], [22]. The uniform reweighting techniques include reweighting based on inverse class frequency [6], [23] or based on the effective number of samples in each class [6]. On the other hand, fine-grained approaches include Focal loss [22], which reweights based on sample hardness or recent studies [15]–[17] that employ meta-learning to perform sample reweighting.

Data resampling: These methods either repeatedly sample examples from the minority class (over-sampling) [18], [24]–[26] or discard samples from the majority class (under-sampling) [27]–[30]. Popular strategies include SMOTE that over-samples the minority class through linear interpolation [24]; or [27] that under-samples the majority class by clustering and replacing the majority class examples by a few anchor points. With neural networks, over-sampling generally creates redundancy and risks over-fitting to the rare classes, while, under-sampling is susceptible to losing information from the majority classes [9].

Training strategies: These methods modify the training procedure to mitigate the problem of class imbalance. For instance, LDAM [21], introduces a delayed reweighting scheme wherein, class reweighting is applied after a few epochs of training. Kang et al. [7] show improvement through a two-step training process which decouples representation and classifier learning. Recently, BBN [31] show that gradually shifting emphasis from class sampling to reverse sampling helps improve accuracy.

`HAR` is complementary to the above three directions in that it introduces a hardness aware learning framework that readily works with many existing loss reweighting and training strategies to further improve accuracy on imbalanced datasets.

B. Hardness aware learning methods

The concept of example hardness is central to a variety of successful learning techniques including those from curriculum learning [32], [33] (which deals with ordering the training set in increasing order of hardness), hard negative mining [13], [22], [34] (which deals with identifying false positive samples in order to improve classification performance) and reinforcement learning [35], [36] (which deals with ordering tasks from easy to hard in order to aid an agent's learning). The underlying assumption in these methods is that "harder examples" contribute a stronger learning signal and are more informative than their "easier" counterparts [13], [14]. To this end, example hardness can be defined in multiple ways. For example, OHEM [34] uses absolute loss as an indicator of hardness; focal loss [22] uses the prediction-confidence of true class probability as a measure of hardness; Hacohen et. al. [33] use the prediction-confidence of a pre-trained teacher model as the measure of hardness. Differing from these, `HAR` uses a more general hardness measure which embodies the intuition that easy examples are those which can be confidently and correctly predicted using coarse-features from earlier layers of a neural network. This enables `HAR` to impart hardness awareness to many existing loss functions.

C. Multi-branch neural networks

Research in this area aims to endow a neural network with auxiliary classifier branches (or early-exits) that allow for obtaining predictions from intermediate locations along the backbone DNN. This leads to advantages such as, saving inference time compute [37]–[39]; mitigating the vanishing gradient problem as in Inception networks [40]; while also affording a natural application to computing paradigms such as fog computing and 5G [41], [42]. The important research challenges for multi-branch DNNs (see review paper [43]) stem from questions such as: where to place the early-exits, what criterion to use for early-exiting and how to define the training objective. Many works place the early-exits after each block of layers which enables large savings of inference FLOPS [37]–[39]. The exit criteria range from early-exiting based on low entropy predictions as in BranchyNet [44] to early exiting based on prediction confidence as in MSDNet [37]. Training objectives include ones that distill knowledge from later exits to earlier ones [38], [45] or ensemble predictions from multiple branches to improve adversarial robustness [39]. To the best of our knowledge, `HAR` is the first to use multi-branch DNNs to enable hardness-aware loss reweighting for long-tailed imbalance. Our experiments show this leads to large accuracy improvement in the presence of class imbalance.

III. Methodology

A. Why care about hardness?

We hypothesize that within both the majority and minority classes some examples are easier to classify than others. Consequently, not every example in the minority class needs to be equally upweighted; and not every example in the majority class needs to be equally downweighted. In order to verify this hypothesis, we train a ResNet-32 model on CIFAR-10 LT (using vanilla cross entropy) and determine: What proportion of the rarest class examples obtain a high confidence prediction (≥ 0.9) and what proportion of the majority class examples obtain a low confidence prediction ($\leq$0.1). Fig. 3 plots outcome of this experiment.

As expected, many examples from the minority class are classified with low confidence while many examples from the majority class are predicted with near certainty. However, confirming our hypothesis, a considerable proportion of the majority class examples obtain a low confidence prediction

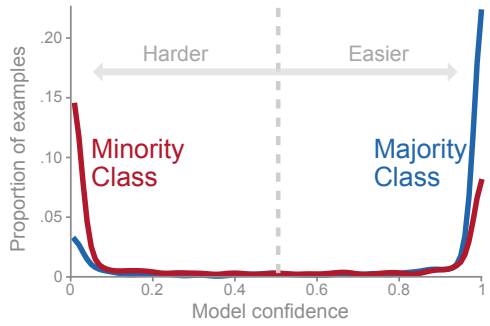


Fig. 3: Observe that a considerable proportion of the majority class examples predicted by a ResNet-32 trained on CIFAR-10 LT using cross entropy, obtain a low confidence prediction and vice versa. It is precisely this subset of examples—*low confidence majority* and *high confidence minority*—that HAR impacts the most. Particularly, HAR increases the loss contribution of low confidence *majority* examples while retaining the original loss contribution for high confidence *minority*.

and vice versa. It is precisely this subset of examples—*low confidence majority* and *high confidence minority*—that HAR impacts the most. In particular, HAR increases the loss contribution of low confidence *majority* examples while retaining the original loss contribution for high confidence *minority* examples thereby enabling a fine-grained, hardness aware approach to loss reweighting.

Problem Setup. Denote a multi-exit neural network by $f : \boldsymbol{X} \rightarrow \boldsymbol{z}$ that maps an input example $\boldsymbol{X} \in \mathbb{R}^{h \times w \times 3}$ to a list of prediction vectors $\boldsymbol{z} = [\boldsymbol{z}^{(1)}, ..., \boldsymbol{z}^{(k)}]$ where the vector $\boldsymbol{z}^{(k)} \in \mathbb{R}^c$ specifies the prediction confidence over c classes at the k^{th} exit. Assuming the weights $\boldsymbol{\theta}^{(k)}$ parameterize f up to exit k, then we have $\boldsymbol{z}^{(k)} = f(\boldsymbol{X}; \boldsymbol{\theta}^{(k)})$ as the output at the k^{th} exit. The supervised learning task specifies training f on a training dataset $\mathcal{D} = \{(\boldsymbol{X}_1, y_1), ..., (\boldsymbol{X}_n, y_n)\}$ containing c classes. If n_j denotes the number of training examples in class j, then a long tail dataset satisfies $n_i > n_j, \forall i < j$ and $n_1 >> n_c$. Typically, post training, f is evaluated on a balanced test set $\mathcal{D}'$ which satisfies $n'_i = n'_j, \forall i, j$. The goal of imbalanced classification is to maximize an evaluation metric such as top-1 accuracy on $\mathcal{D}'$ while learning a classifier on $\mathcal{D}$.

B. Training a multi-branch DNN with HAR

1) ***Preliminaries****:* Training algorithms for multi-branch neural networks come in many shapes and forms. The most popular ones [37], [39], [44], [46], [47] train all branches simultaneously by defining a single training objective that incorporates the predictions from all intermediate branches. One way to accomplish this [37], [44], [46] is to define the aggregated loss as a weighted sum of the loss computed at each exit

$$\mathcal{L}_{multi-exit}(\boldsymbol{z}_i, y_i) = \sum_{k=1}^{K} \alpha_k \mathcal{L}(\boldsymbol{z}_i^{(k)}, y_i), \tag{1}$$

where $\boldsymbol{z}_i = [\boldsymbol{z}_i^{(1)}; ..., \boldsymbol{z}_i^{(K)}]$ is the list of predictions from K branches, $\alpha_k$are positive constants, and $\mathcal{L}$ is a classification loss such as the cross-entropy. Another strategy [47] is to first combine the prediction vectors at each branch

$$\hat{\boldsymbol{z}}_i = \sum_{k=1}^{K} \alpha_k \boldsymbol{z}_i^{(k)}, \tag{2}$$

and then obtain the training loss as $\mathcal{L}_{multi-exit}(\boldsymbol{z}_i, y_i) = \mathcal{L}(\hat{\boldsymbol{z}}, y_i)$. Among these two approaches, HAR's training objective is inspired from the former (*i.e.*,Eq. 1).

2) ***General HAR loss****:* The goal of the HAR training objective is to up weight the loss contribution of hard examples. To this end, we modify Eq. (1) in two ways. First, we simplify the hyperparameter design space by setting $\alpha_k = 1$ (thus weighing all branches equally). Second, we introduce conditional aggregation into the summation term on the right hand side with the general idea: to aggregate the loss up until the first exit where the neural network *correctly* and *confidently* predicts the class of an input example. This objective is defined as

$$\begin{aligned} \mathcal{L}_{\text{HAR}}(\boldsymbol{z}_i, y_i) &= \sum_{k \in [1,...,k_i]} \mathcal{L}\left(\boldsymbol{z}_i^{(k)}, y_i\right), \\ \text{where } k_i &= \underset{j \in \{1,2,...,K\}}{\text{argmin}} \left(g_i^{(j)} > 0\right). \end{aligned} \tag{3}$$

Here $g_i^{(j)}$ defines the early exiting criterion for example i at exit j and is defined as below

$$g_i^{(j)} = \begin{cases} 1, & \text{if } \text{argmax}(\boldsymbol{z}_i^{(j)}) = y_i \text{ and } \boldsymbol{z}_i^{(j)}[y_i] > t \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

The sum in Eq.(3) means that an example $\boldsymbol{X}_i$ accumulates an exit loss $\mathcal{L}\left(\boldsymbol{z}_i^{(k)}, y_i\right)$ up to the first exit k_i where it exits by satisfying the exit criterion of Eq. (4). Further, the exit criterion of Eq. (4) is satisfied (*i.e.*,outputs a 1) only when the prediction at exit k is both: *correct* (*i.e.*,$\text{argmax}(\boldsymbol{z}_i^{(j)}) = y_i$) and *confident* (*i.e.*,$\boldsymbol{z}_i^{(j)}[y_i] > t$). Thus viewed together, Eqs. (3) and (4) ensure that hard examples exit later by virtue of which, encumber a larger loss.

3) ***Instantiating the HAR loss****:* The HAR training objective of Eq.(3) is agnostic to the exact instantiation of $\mathcal{L}$ at branch k. In particular, $\mathcal{L}$ can be any loss function useful for class imbalance, including: weighted cross-entropy [6], Focal loss [22], LDAM loss [21] or any combination thereof. We conduct experiments with both weighted cross entropy and the recently proposed LDAM loss as the exit loss type.

When using the class weighted cross-entropy at each exit, the HAR training objective is described as follows

$$\mathcal{L}_{\text{HAR}}^{\text{CE}}(\boldsymbol{z}_i, y_i) = \sum_{k \in [1,...,k_i]} \boldsymbol{w}_{y_i} log\left(\frac{exp(\boldsymbol{z}_i^{(k)}[y_i])}{\sum_{j=1}^{C} exp(\boldsymbol{z}_i^{(k)}[j])}\right), \tag{5}$$

where $\boldsymbol{w}_{y_i}$ refers to the class specific weight, which according to prior work, can be set based on the inverse class frequency [10], [11], inverse square root frequency [12], [48] or the effective

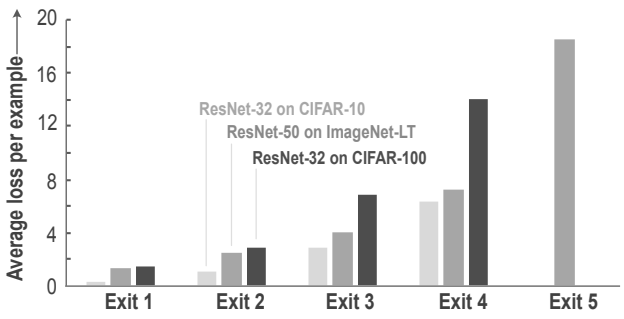


Fig. 4: On multiple models (ResNet-32/50) and datasets (CIFAR LT/ImageNet LT) we observe that examples exiting later indeed contribute a higher average loss per sample. This figure empirically validates Property 1.

weighting [6] strategies. This work leverages the weighting strategy from [6] which sets $\boldsymbol{w}_c = \frac{1-\beta}{1-\beta^{n_c}}$, where n_c is the number of samples in class c and β is a hyperparameter with typical values in $\{0.999, 0.9999\}$.

When using LDAM [21] at each exit, the HAR loss is described as

$$\begin{aligned}\mathcal{L}_{\text{HAR}}^{\text{LDAM}}(\boldsymbol{z}_i, y_i) &= \sum_{k\in[1,\ldots,k_i]} \boldsymbol{w}_{y_i} log\left(\boldsymbol{p}_i^{(k)}\right) \\ \text{where, } \boldsymbol{p}_i^{(k)} &= \frac{exp(\boldsymbol{z}_i^{(k)}[y_i] - \triangle_{y_i})}{exp(\boldsymbol{z}_i^{(k)}[y_i] - \triangle_{y_i}) + \sum_{j\neq y_i} exp(\boldsymbol{z}_i^{(k)}[j])} \\ \text{and } \triangle_{y_i} &= \frac{C}{n_{y_i}}\end{aligned} \tag{6}$$

The quantity $\triangle_{y_i}$ defines a per-class margin that ensures rare classes get a larger margin. It is determined by a hyperparameter C and the number of examples n_{y_i} in class y_i. Following [21], we select C such that the largest margin for any class is 0.5.

*4) **Outcome of training with the HAR loss**:* The intended goal of HAR loss is to increase the loss contribution of hard examples. This is formally stated in the following property.

Property 1: (Increasing Loss Property) For a multi-exit neural network f, if D_k denotes the set of examples exiting at exit k then, $\forall i < j$, $\mathbb{E}_{(\boldsymbol{z}_m, y_m)\in D_i}[\mathcal{L}_{\text{HAR}}(\boldsymbol{z}_m, y_m)] < \mathbb{E}_{(\boldsymbol{z}_m, y_m)\in D_j}[\mathcal{L}_{\text{HAR}}(\boldsymbol{z}_m, y_m)]$.

The above property simply states that the average training loss for examples exiting at exit *i*, monotonically increases across exits. This enables the neural network to focus on harder examples (which contribute a higher expected loss) from both the minority and majority classes. To empirically validate this property, in Fig. 4, we plot the average training loss contributed by examples exiting at each auxiliary branch for ResNet-32 and ResNet-50 models trained with $\mathcal{L}_{\text{HAR}}^{CE}$ in Eq. 5, on the CIFAR-10/100 and ImageNet LT datasets. The increasing trend of average loss across exits indeed validates that examples exiting later do contribute a higher loss.

C. Inference in multi-branch neural networks

*1) **Preliminaries**:* During inference, the outputs of a multi-branch DNN can be aggregated into a single prediction vector in several ways. The popular choices include (1) Inception networks [40] which discard the branch predictions and use only the backbone output as the overall prediction, (2) Hu et al. [39] which uses the mean of all branch outputs as the overall prediction or (3) MSDNet [37] which selects a branch based on prediction confidence, whose output is chosen as the overall prediction. With HAR, a model can support two modes of inference corresponding to choices (1) and (3) above. We term (1) as the *static mode* and (3) as the *dynamic mode*. The next subsection dives deeper into the dynamic mode.

*2) **Dynamic inference with HAR**:* Similar to MSDNet [37], HAR selects a branch (based on prediction confidence), who's output is designated as the network prediction. More formally, given the multi-branch prediction $\boldsymbol{z}_i = [\boldsymbol{z}_i^{(1)}, ..., \boldsymbol{z}_i^{(K)}]$ for an input $\boldsymbol{X}_i$, the overall prediction $\hat{\boldsymbol{z}}_i$ of the network is

$$\hat{\boldsymbol{z}}_i = \boldsymbol{z}_i^{(k_i)} \text{ , where } k_i = \underset{j\in\{1,\ldots,K\}}{\text{argmin}} \left(h_i^{(j)} > 0\right), \tag{7}$$

where $h_i^{(j)}$ is the prediction confidence for example i at exit j

$$h_i^{(j)} = \begin{cases} 1, & \text{if } \text{argmax}(\boldsymbol{z}_i^{(j)}) > s \\ 0, & \text{otherwise} \end{cases} \tag{8}$$

Intuitively, Eq. 7 specifies the overall network prediction as the output of the earliest exit where the exit-criterion of Eq. 8 is satisfied. Further, the exit criterion of Eq. 8 is satisfied when the prediction confidence (returned by the *argmax*) exceeds a predefined threshold s. Thus, the dynamicity during inference arises out of varying the value of s which affects the exit criterion in the following way: a lower value of s leads to a relaxed version of hardness, and thus more early exiting. Or in other words, s is a control knob for dynamically controlling the inference FLOPS of a network.

IV. Experiments

In this section, we begin by discussing the experimental setup including: (i) datasets, (ii) evaluation metrics (iii) backbone models and training hyperparameters, (iv) training configurations, and (v) hyperparameter search. Following this, we answer the following three questions:

1) Does HAR benefit in both modes of inference? (Sec. IV-C)
2) How does HAR compare to the state of the art? (Sec. IV-D)
3) What is the class-composition of early exiting examples in the dynamic mode? (Sec. IV-E)

Finally, we end the section with an ablation study on different modelling choices of HAR (Sec. IV-F).

A. Experimental Setup

Datasets. We conduct our evaluation on four long-tailed datasets: CIFAR-10 LT, CIFAR-100 LT [6], ImageNet LT [2] and iNaturalist'18 [20]. For the first three datasets (CIFAR

LT & ImageNet LT), the training split is obtained by sub-sampling from their balanced versions. In case of the CIFAR LT datasets, we consider three levels of imbalance, $10\times$, $50\times$ and $100\times$, which is defined as the ratio between the number of samples in the largest and the smallest classes. The ImageNet LT training split consists of 115.8k images from 1,000 classes with largest and smallest classes containing 1,280 and 5 images respectively. The iNaturalist'18 dataset is a naturally imbalanced dataset containing 437,513 training images from 8,142 species of plants and animals. For the ImageNet LT and iNaturalist datasets, similar to [7] we present results on the *many-shot* (classes containing >100 examples), *medium-shot* (classes containing 20-100 examples), and *few-shot* (classes containing <20 examples) splits. Please refer to Appendix A for additional details on the dataset construction.

Evaluation Metrics. We follow the same evaluation setting as recent methods [6], [7], [21], [31]. For all datasets, the training split is imbalanced (See appendex A) while the validation and test splits are balanced. The top1 accuracy on the test split serves as the common metric of comparison across all datasets.

Backbone models & training configurations. We consider several models from the ResNet and DenseNet families. To obtain an augmented HAR model, we attach auxiliary classifier branches before each residual/dense block (see Appendix B for details). On CIFAR datasets, we train the augmented ResNet-32 models for 200 epochs using SGD with an initial learning rate of 0.1 decreased by 0.01 at epochs 160 and 180 [6], [21]. The weight decay is 2×10^{-4}. On ImageNet LT and iNaturalist'18 we train the augmented ResNet-50 and DenseNet-169 models for 100 epochs using SGD with an initial learning rate of 0.1 decreased by 0.1 at epochs 60 and 80. The weight decay is 2×10^{-4}. Similar to [6], [21], all models use a linear warm-up schedule for the first 5 epochs to avoid initial over-fitting.

Implementation. HAR is constructed from three *key* components—(1) an exit loss function, (2) a class reweighting strategy, and (3) a reweighting schedule. For the exit loss, we consider two variations with HAR—at each exit we use either cross entropy loss (CE) or label distribution aware margin loss (LDAM) [21]. These are referred to as HAR(CE) and HAR(LDAM). For class reweighting, we weight each example of class c according to it's effective number $\frac{1-\beta}{1-\beta^{n_c}}$, where n_c is the number of images in class c [6]. Finally, for the reweighting schedule we use the per-dataset delayed reweighting (DRW) scheme introduced in [21]. All experiments are conducted on a system with four V100 GPUs and 512GB of RAM.

Baselines. To measure the performance of HAR, we compare against three strong baselines: CE [6], Focal [22] and LDAM [21], each reusing the same class reweighting and delayed reweighting schedule discussed above. For Focal loss we set γ = 0.5 [22], and for LDAM we set C such that the maximum margin is 0.5 [21].

B. Hyperparameter Search for HAR

HAR introduces two hyperparameters—training and inference exit thresholds t, s in Eq. (4),(8) respectively. Parameter t controls the number of early exiting examples *during training*. Smaller values result in many examples exiting early leading to a more relaxed definition of example hardness. Parameter s, is a control knob for varying the computational cost during inference. We determine s directly on the test split depending on the desired FLOPS saving, while the optimal value of t is determined through a line search on the validation split. Tab. I presents the top-1 accuracy on the three (many/med/few shot) splits of ImageNet LT, reached by a ResNet-50 model trained using different values of t.

Selecting t for HAR(CE) For exit loss type cross entropy, we consider a line search in [0.75,0.95] in steps of 0.05. Tab. Ia shows the top-1 accuracy peaks when $t = 0.9$.

Selecting t for HAR(LDAM) For exit loss type LDAM [21], we consider a line search in [1.7/|c|, 2.1/|c|] in steps of 0.1/|c| where |c| is the number of classes in the dataset. Tab. Ib shows the top-1 accuracy peaks when $t = 2.0/1k$.

We reuse the above hyperparameters: t=0.9 for HAR(CE), and $t = 2.0/|c|$ for HAR(LDAM); on all datasets.

TABLE I: Line search for inference threshold t in HAR. We observe a clear trend wherein the top-1 validation accuracy (of a ResNet-50 trained on ImageNet LT) increases with larger t until a certain point after which it falls. The column with the optimal value of t is highlighted in gray.

t	0.75	0.8	0.85	0.9	0.95
Many	60.2	60.8	61.1	62.4	62.6
Med	44.6	45.0	45.6	47.3	46.1
Few	25.3	26.0	26.4	28.5	28.1
All	48.0	48.5	48.9	50.5	50.0

(a) Identifying t for HAR(CE)

t	1.7/1k	1.8/1k	1.9/1k	2.0/1k	2.1/1k
Many	59.7	63.2	65.0	65.7	65.5
Med	47.9	48.5	48.4	48.2	47.8
Few	31.2	30.1	30.1	29.9	28.8
All	50.2	51.7	52.3	52.5	52.0

(b) Identifying t for HAR(LDAM)

C. Two inference modes of HAR

I. Static mode of inference refers to the state when the bare backbone model, without any early exits, is used for inference. Such a model engenders a fixed compute capacity (or FLOPS) during inference. Fig. 5 plots the top-1 accuracy *vs*. the inference FLOPS for ResNet-50/DenseNet-169 models trained on ImageNet LT. The solid red squares and solid blue triangles depict the static DenseNet-169 and ResNet-50 models trained with different loss functions. Observe that the static models trained with HAR outperform other loss functions indicating the merits of enabling hardness-awareness during training.

II. Dynamic mode of inference refers to the state when the intermediate branches are preserved (during inference) and used to early-exit easier examples. In this mode, the compute capacity (or FLOPS) can be varied by changing the inference threshold parameter s in Eq. 8. Each value of s leads to a new

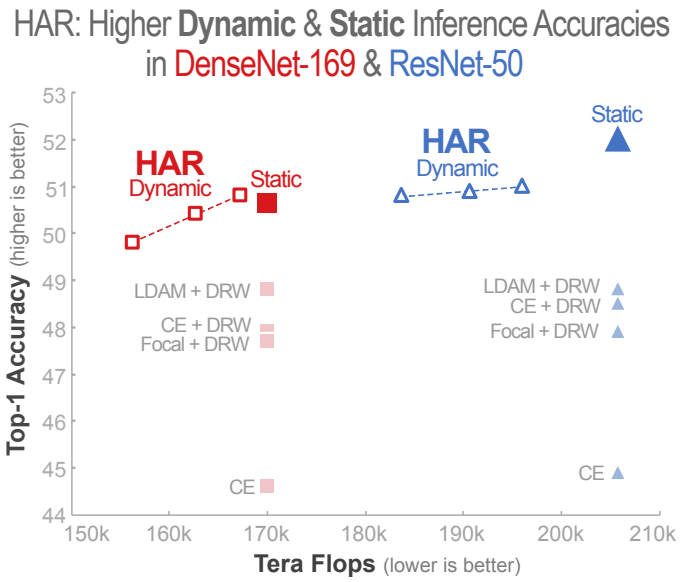


Fig. 5: The static DenseNet-169 (red squares) and ResNet-50 (blue triangle) trained on ImageNet LT lie along a vertical line and correspond to a *fixed* FLOPS budget. In contrast, the dynamic models trained with HAR lie along an accuracy *vs.* efficiency trade-off curve. Observe that HAR leads to higher accuracy for both static and dynamic modes while additionally leading to FLOPS savings in the dynamic mode.

point on the accuracy *vs.* efficiency trade-off curve. The hollow red squares and blue triangles in Fig. 5 plot this trade-off for a DesneNet-169 and ResNet-50 model trained with HAR. The three points are obtained by setting $s \in \{1.7, 1.75, 1.8\} \times 10^{-3}$. We observe that the accuracy-efficiency curve envelopes the other baselines indicating that HAR leads to a favorable trade-off even in the dynamic mode.

D. Comparison to the state-of-the-art

We compare against the s.o.t.a, assuming the dynamic mode of inference for HAR, since it leads to practical compute savings. We observe that HAR is able to improve the accuracy of s.o.t.a methods (CE+DRW & LDAM+DRW) by incorporating the notion of hardness during training. Further discussion is driven by three broad questions.

I. How does HAR perform under different levels of imbalance? To answer this question, we follow prior work [21], [31] and train a ResNet-32 model on CIFAR-10 LT and CIFAR-100 LT with three imbalance levels $(100\times, 50\times, 10\times)$. The results in Tab. II present two key observations.

1) **HAR improves accuracy under all imbalance levels** $(100\times,\ 50\times,\ 10\times)$ for both exit-loss types (HAR(CE) and HAR(LDAM)) while consuming $15-30\%$ fewer FLOPS on CIFAR-10 and $2-10\%$ fewer FLOPS on CIFAR-100.
2) **Accuracy gap increases with imbalance**. The accuracy improvement is higher for greater levels of imbalance (*e.g.*,1.1%, 1%, 0.4% improvement over LDAM for $100\times, 50\times, 10\times$ on CIFAR-10)

II. Which classes drive the overall improvement in accuracy? To answer this question, we follow prior work [7] to train a ResNet-50 model on ImageNet LT and iNaturalist-18 datasets. In Tab. III, for each method, we dissect its overall top-1 accuracy into accuracies on three class splits: *many-shot* classes (>100 examples/class), *medium-shot* classes (20-100 examples/class), and *few-shot* classes (<20 examples/class). Following are the key observations:

1) **Many-shot split drives accuracy improvement**. On both datasets, we observe that the greatest accuracy improvements are observed on the many-shot splits (*e.g.*,$+2.7\%$ relative to LDAM on ImageNet LT, and $+2.2\%$ on iNaturalist'18).

TABLE II: HAR leads to highest top-1 accuracies while saving inference FLOPS, for ResNet-32 models trained on long tailed CIFAR-10 and CIFAR-100 datasets. We consider three levels of imbalance $(100\times, 50\times, 10\times)$. Top rows are recent methods; middle and bottom rows compare HAR fitted with two different exit-loss types (CE/LDAM). Our re-implementation marked by †; results from [31] marked by ††.

	CIFAR-10 LT			CIFAR-100 LT		
	100×	50×	10×	100×	50×	10×
CE†	70.4	74.8	83.6	28.3	43.9	55.7
Focal [22]††	70.4	76.7	86.7	28.4	44.3	55.8
Mixup [49]††	73.1	77.8	87.1	39.5	45.0	58.0
Manifold Mixup [50]††	73.0	78.0	87.0	38.3	43.1	56.5
CE+DRW [6]†	76.3	80.0	87.6	41.4	46.0	58.3
HAR(CE)+DRW (Our)	**76.8**	**80.8**	**87.6**	**42.5**	**47.1**	**58.7**
Flops saving	32%	29%	26%	11%	11%	10%
LDAM+DRW [21]†	77.0	81.4	87.6	42.0	46.6	58.7
HAR(LDAM)+DRW (Our)	**78.1**	**82.4**	**88.0**	**43.1**	**47.5**	**58.9**
Flops Saving	15%	21%	20%	0%	2%	3%

TABLE III: HAR leads to highest top-1 accuracies with compute savings, for ResNet-50 trained on Imagenet LT and iNaturalist'18 datasets. We consider the many-, medium-, and few-shot splits. Top rows are recent methods; middle and bottom rows compare HAR fitted with two different exit-loss types (CE/LDAM). Our re-implementation marked by †; results from [7] marked by ††.

	ImageNet LT				iNaturalist'18			
	Mny	Med	Few	All	Mny	Med	Few	All
CE†	63.8	38.5	13.6	44.6	72.7	63.8	58.7	62.7
CRT [7]††	58.8	44.0	26.1	47.3	69.0	66.0	63.2	65.2
LWS [7]††	57.1	45.2	29.3	47.7	65.0	66.3	65.5	65.9
τ-norm [7]††	56.6	44.2	27.4	46.7	65.6	65.3	65.9	65.6
Focal+DRW [22]†	59.5	44.6	27.0	47.9	66.1	66.0	64.3	65.4
CE+DRW [6]†	60.3	45.2	27.0	48.5	67.1	66.2	**65.4**	65.9
HAR(CE)+DRW (Our)	**60.7**	**45.5**	**27.7**	**48.9**	**67.4**	**66.3**	65.1	**66.0**
Flops Saving				21%				13%
LDAM + DRW [21]†	61.1	44.7	28.0	48.8	70.0	67.4	**66.1**	67.1
HAR(LDAM)+DRW (Our)	**63.8**	**47.2**	**28.1**	**51.0**	**72.2**	**69.0**	65.7	**68.0**
Flops Saving				5%				2%

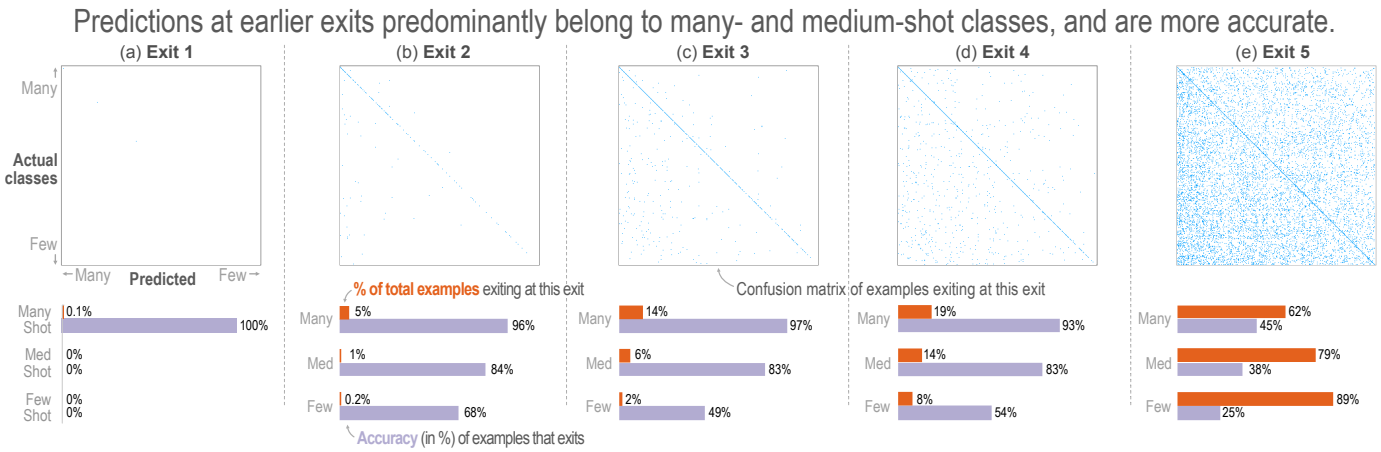


Fig. 6: For a ResNet-50 trained on ImageNet LT with HAR (LDAM)+DRW, we show at each exit: (*top row*) The confusion matrix of predictions, (*bottom row*) The percentage of total examples for each split (many-, medium-, few-shot) that exit and the accuracy of these predictions. We note that (1) the early exiting examples are predominantly from the many- and medium-shot classes (2) the predictions at early exits, exhibit a high accuracy.

2) **Baseline methods lose significant accuracy on many- and medium-shot classes**. Relative to the CE baseline, we observe SOTA methods lose significant accuracy on the many- and medium-shot classes. This explains how HAR improves overall accuracy: It mitigates the accuracy decline on majority and medium shot classes while leading to comparable accuracy gain on the few-shot classes.

E. Analyzing the dynamic inference mode

I. Which classes tend to exit earlier? To answer this question, in Fig. 6, we present the confusion matrix and the class-composition for each exit of a ResNet-50 trained on ImageNet LT with HAR. Following are some key observations.

1) **Early-exiting examples are predominantly from the majority classes.** The confusion matrices in the top row show a distribution shift from a top-left bright diagonal in part a (exit-1) to a middle heavy diagonal in part d (exit 4). Since the 1000 classes are sorted according to size such that top and left contain the majority classes while bottom and right contain the rare classes, this observation indicates that the initial exits prefer the many-shot classes while the later exits prefer the medium-shot classes. The red bar plots of Fig. 6a-e reinforce this observation by showing the percentage of each split that has exited by each exit. Notice that before last exit, 38% of the many-shot, 21% of the medium-shot and 11% of the few-shot split have already been classified.
2) **Early exiting examples are more trustworthy due to higher accuracy.** The purple bar plots of Fig. 6a-e present the accuracy for examples exiting at each exit. Notice that for all three splits (many, med, few shots), the accuracy for exits 1-4 is nearly double than that of exit-5. This indicates that an early prediction is much more trustworthy (due to higher accuracy) than that a late prediction.

TABLE IV: HAR leads to the highest top-1 accuracy for a ResNet-50 model trained on ImageNet LT, and, without any class re-weighting. We consider the static inference mode.

Loss	Many	Med	Few	All
Focal ($\gamma = 0.5$)	63.5	38.5	13.6	44.7
Focal ($\gamma = 1.0$)	62.8	37.1	12.7	43.7
Focal ($\gamma = 2.0$)	62.1	36.6	11.9	43.1
CE	63.8	38.5	13.6	44.9
HAR (CE)	**63.9**	**39.9**	**16.0**	**45.9**
LDAM	64.9	39.1	12.6	45.5
HAR (LDAM)	**66.3**	**42.4**	**14.2**	**47.8**

TABLE V: Ablating on the location of a single early exit (indicated by a E) using a *static* ResNet-50 trained with HAR (LDAM)+ DRW on ImageNet LT. The naming convention specifies C for a convolution layer, B for a residual block and E for an early exit. Configuration CBBEBBE outperforms others, suggesting the value of an early exit peaks between blocks 2,3.

Model	Many	Med	Few	All
CBBBBE	61.1	44.7	28.0	48.8
CEBBBBE	62.3	44.9	25.9	49.0
CBEBBBE	63.0	45.9	26.7	49.9
CBBEBBE	**63.3**	**46.6**	**27.8**	**50.5**
CBBBEBE	62.5	46.0	27.6	49.8

F. Ablation studies

How does HAR perform without class reweighting? Tab. IV presents the accuracy of a ResNet-50 model trained on ImageNet LT with three different loss functions—focal, cross entropy and LDAM—and without any class reweighting at the early-exits. We observe that the (static) models trained with HAR outperform the vanilla loss functions (CE, LDAM) on all the three splits.

What is the impact of the location of early exits? To measure this, in Tab. V, we ablate on the location of a single early exit

Fig. 7: Images exiting from the first, third and the final exit of a HAR model indicate that as the exits increase, so does the visual hardness.

attached to a ResNet-50 model trained on ImageNet LT using the HAR (LDAM)+DRW loss. For comparison, we use the static mode, which means the exits are only used during training and are discarded for inference. Among the several configurations (with naming convention: C=conv layer, B=block, E=exit), the one with an exit after block 2—CBBEBBE—outperforms all others. This suggests a diminishing value of placing an exit too early/late in the backbone.

Can we visualize the learned notion of example hardness? Fig. 7 presents a randomly selected subset of test split images exiting through the auxiliary branches of a HAR ResNet-50 model trained on ImageNet LT. Among each class, we observe that the object of interest is easier to distinguish in images exiting from earlier branches which indicates that HAR enables a model to learn an intuitive notion of image hardness.

V. Conclusion

We identified the notion of *sample-hardness* as a key concept to improve generalization under a long-tailed class distribution. To incorporate this notion of hardness in the learning process, we proposed the HAR framework. HAR is complementary to existing work in long-tailed classification and can readily integrate with existing approaches to improve classification accuracy. Extensive evaluations demonstrate that HAR outperforms existing state-of-the-art techniques while saving inference FLOPS.

Appendix A
Dataset Construction

We follow prior work [6], [7], [21], [31] for constructing the datasets. Specifically, the train split is subsampled as follows, while the val/test split is left class-balanced.

CIFAR LT datasets. Following [6], the train sets for CIFAR-10 LT, CIFAR-100 LT are sampled from the original training sets of CIFAR-10 and CIFAR-100 according to the exponential distribution $n_c = n\mu^c$. Here n_c refers to the remaining number of examples in class c, n is the original number of examples per class (5000 for CIFAR-10 and 500 for CIFAR-100) and $\mu \in [0, 1]$. We select μ such that the imbalance ratio—which is defined as the ratio between the number of examples in the largest and smallest class—is $10\times, 50\times, 100\times$.

ImageNet LT dataset. Following [2], the training set is sub-sampled from the original ImageNet training set by following the pareto distribution with $\alpha = 6$. The val split is the same as the original ImageNet dataset.

iNaturalist'18 dataset [20]. This is a naturally imbalanced dataset consisting of images from 8,142 species. The validation and test splits are balanced across classes.

Appendix B
Architecture of HAR models

HAR attaches an auxiliary exit before each residual/dense block. The augmented models are shown in Fig. 8, with the auxiliary exit design considerations discussed below.

ResNet-32: This backbone model contains three residual block groups (see Fig. 8a), with each group containing five standard "basic blocks". Each auxiliary exit consists of two convolution layers with sixty-four kernels of size 3×3, followed by an average pooling and dense layer.

ResNet-50: This backbone model contains four residual block groups (see Fig. 8b), with the groups containing $3, 4, 6$ and 3 "bottleneck" blocks respectively. Since the filter channels increase rapidly in this architecture (e.g. group three has 1024 channels), we use depth-wise separable convolution layers at each exit which helps reduce the additional FLOPS introduced by the auxiliary exits.

DenseNet-169 This backbone model contains four dense block groups (see Fig. 8c), with the groups containing $6, 12, 32$ and 32 "dense" blocks respectively. Each auxiliary exit consists of two convolution layers with 3×3 kernels followed by an average pooling and dense layer.

References

[1] M. A. Mazurowski, P. A. Habas, J. M. Zurada, J. Y. Lo, J. A. Baker, and G. D. Tourassi, "Training neural network classifiers for medical decision making: The effects of imbalanced datasets on classification performance," *Neural networks*, 2008.

[2] Z. Liu, Z. Miao, X. Zhan, J. Wang, B. Gong, and S. X. Yu, "Large-scale long-tailed recognition in an open world," in *CVPR*, 2019.

[3] R. Duggal, S. Freitas, C. Xiao, D. H. Chau, and J. Sun, "Rest: Robust and efficient neural networks for sleep monitoring in the wild," in *Proceedings of The Web Conference 2020*, 2020, pp. 1704–1714.

[4] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," 2009.

[5] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *CVPR*, 2009.

[6] Y. Cui, M. Jia, T.-Y. Lin, Y. Song, and S. Belongie, "Class-balanced loss based on effective number of samples," in *CVPR*, 2019, pp. 9268–9277.

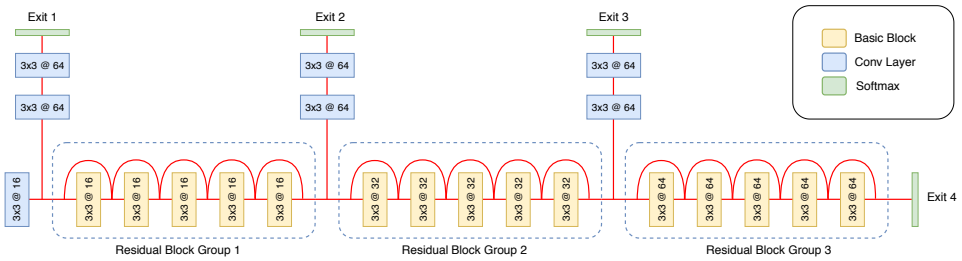
[7] B. Kang, S. Xie, M. Rohrbach, Z. Yan, A. Gordo, J. Feng, and Y. Kalantidis, "Decoupling representation and classifier for long-tailed recognition," *ICLR*, 2020.

[8] P.-H. C. Chen, Y. Liu, and L. Peng, "How to develop machine learning models for healthcare," *Nature materials*, 2019.

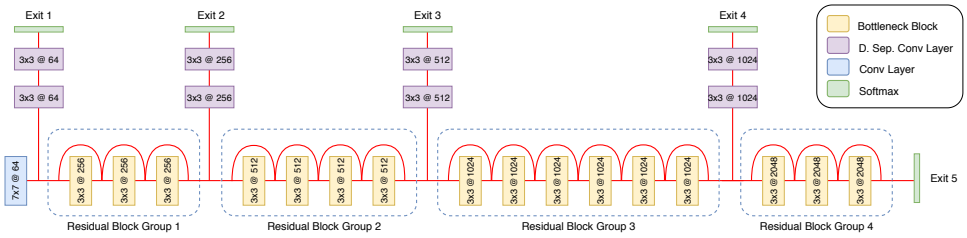
[9] M. Buda, A. Maki, and M. A. Mazurowski, "A systematic study of the class imbalance problem in convolutional neural networks," *Neural Networks*, vol. 106, pp. 249–259, 2018.

[10] C. Huang, Y. Li, C. Change Loy, and X. Tang, "Learning deep representation for imbalanced classification," in *CVPR*, 2016.

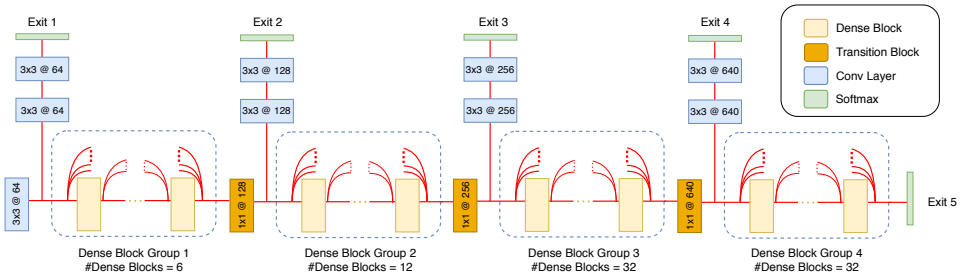
[11] Y.-X. Wang, D. Ramanan, and M. Hebert, "Learning to model the tail," in *NIPS*, 2017, pp. 7029–7039.



(a) ResNet-32 model with early exits.



(b) ResNet-50 model with early exits.



(c) DenseNet-169 model with early exits.

Fig. 8: HAR augments a backbone model with auxilliary exits. This figure describes the configuration of the early exits for the three models considered in this work. The notation $3 \times 3@16$ indicates that the block / layer contains 16 kernels of size 3×3.

[12] D. Mahajan, R. Girshick, V. Ramanathan, K. He, M. Paluri, Y. Li, A. Bharambe, and L. van der Maaten, "Exploring the limits of weakly supervised pretraining," in *ECCV*, 2018, pp. 181–196.

[13] Q. Dong, S. Gong, and X. Zhu, "Class rectification hard mining for imbalanced deep learning," in *ICCV*, 2017.

[14] H.-S. Chang, E. Learned-Miller, and A. McCallum, "Active bias: Training more accurate neural networks by emphasizing high variance samples," *arXiv preprint arXiv:1704.07433*, 2017.

[15] M. Ren, W. Zeng, B. Yang, and R. Urtasun, "Learning to reweight examples for robust deep learning," in *ICML*, 2018.

[16] J. Shu, Q. Xie, L. Yi, Q. Zhao, S. Zhou, Z. Xu, and D. Meng, "Meta-weight-net: Learning an explicit mapping for sample weighting," *arXiv preprint arXiv:1902.07379*, 2019.

[17] M. A. Jamal, M. Brown, M.-H. Yang, L. Wang, and B. Gong, "Rethinking class-balanced methods for long-tailed visual recognition from a domain adaptation perspective," in *CVPR*, 2020.

[18] H. He, Y. Bai, E. A. Garcia, and S. Li, "Adasyn: Adaptive synthetic sampling approach for imbalanced learning," in *IJCNN*.

[19] N. V. Chawla, A. Lazarevic, L. O. Hall, and K. W. Bowyer, "Smoteboost: Improving prediction of the minority class in boosting," in *European conference on principles of data mining and knowledge discovery*. Springer, 2003, pp. 107–119.

[20] G. Van Horn, O. Mac Aodha, Y. Song, Y. Cui, C. Sun, A. Shepard, H. Adam, P. Perona, and S. Belongie, "The inaturalist species classification and detection dataset," in *CVPR*, 2018.

[21] K. Cao, C. Wei, A. Gaidon, N. Arechiga, and T. Ma, "Learning imbalanced datasets with label-distribution-aware margin loss," in *NIPS*, 2019, pp. 1565–1576.

[22] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *ICCV*, 2017, pp. 2980–2988.

[23] C. Huang, Y. Li, C. L. Chen, and X. Tang, "Deep imbalanced learning for face recognition and attribute prediction," *TPAMI*, 2019.

[24] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: synthetic minority over-sampling technique," *Journal of artificial intelligence research*, vol. 16, pp. 321–357, 2002.

[25] H. Han, W.-Y. Wang, and B.-H. Mao, "Borderline-smote: a new over-sampling method in imbalanced data sets learning," in *International conference on intelligent computing*. Springer, 2005, pp. 878–887.

[26] S. Barua, M. M. Islam, X. Yao, and K. Murase, "Mwmote–majority weighted minority oversampling technique for imbalanced data set learning," *TKDE*, 2012.

[27] S.-J. Yen and Y.-S. Lee, "Cluster-based under-sampling approaches for imbalanced data distributions," *Expert Systems with Applications*, vol. 36, no. 3, pp. 5718–5727, 2009.

[28] M. Galar, A. Fernández, E. Barrenechea, and F. Herrera, "Eusboost: Enhancing ensembles for highly imbalanced data-sets by evolutionary undersampling," *Pattern recognition*, 2013.

[29] W.-C. Lin, C.-F. Tsai, Y.-H. Hu, and J.-S. Jhang, "Clustering-based undersampling in class-imbalanced data," *Information Sciences*, vol. 409, pp. 17–26, 2017.

[30] C.-F. Tsai, W.-C. Lin, Y.-H. Hu, and G.-T. Yao, "Under-sampling class imbalanced datasets by combining clustering analysis and instance selection," *Information Sciences*, vol. 477, pp. 47–54, 2019.

[31] B. Zhou, Q. Cui, X.-S. Wei, and Z.-M. Chen, "Bbn: Bilateral-branch network with cumulative learning for long-tailed visual recognition," *arXiv preprint arXiv:1912.02413*, 2019.

[32] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, "Curriculum learning," in *ICML*, 2009.

[33] G. Hacohen and D. Weinshall, "On the power of curriculum learning in training deep networks," in *ICML*, 2019.

[34] A. Shrivastava, A. Gupta, and R. Girshick, "Training region-based object detectors with online hard example mining," in *CVPR*, 2016.

[35] C. Florensa, D. Held, X. Geng, and P. Abbeel, "Automatic goal generation for reinforcement learning agents," in *ICML*, 2018.

[36] T. Matiisen, A. Oliver, T. Cohen, and J. Schulman, "Teacher–student curriculum learning," *IEEE transactions on neural networks and learning systems*, vol. 31, no. 9, pp. 3732–3740, 2019.

[37] G. Huang, D. Chen, T. Li, F. Wu, L. van der Maaten, and K. Q. Weinberger, "Multi-scale dense networks for resource efficient image classification," in *ICLR*, 2018.

[38] M. Phuong and C. H. Lampert, "Distillation-based training for multi-exit architectures," in *ICCV*, October 2019.

[39] T.-K. Hu, T. Chen, H. Wang, and Z. Wang, "Triple wins: Boosting accuracy, robustness and efficiency together by enabling input-adaptive inference," *arXiv preprint arXiv:2002.10025*, 2020.

[40] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *CVPR*, 2015.

[41] E. Baccarelli, S. Scardapane, M. Scarpiniti, A. Momenzadeh, and A. Uncini, "Optimized training and scalable implementation of conditional deep neural networks with early exits for fog-supported iot applications," *Information Sciences*, 2020.

[42] F. Nan and V. Saligrama, "Adaptive classification for prediction under a budget," *arXiv preprint arXiv:1705.10194*, 2017.

[43] S. Scardapane, M. Scarpiniti, E. Baccarelli, and A. Uncini, "Why should we add early exits to neural networks?" *Cognitive Computation*, 2020.

[44] S. Teerapittayanon, B. McDanel, and H.-T. Kung, "Branchynet: Fast inference via early exiting from deep neural networks," in *ICPR*. IEEE, 2016, pp. 2464–2469.

[45] H. Li, H. Zhang, X. Qi, R. Yang, and G. Huang, "Improved techniques for training adaptive deep networks," in *ICCV*, 2019, pp. 1891–1900.

[46] C.-Y. Lee, S. Xie, P. Gallagher, Z. Zhang, and Z. Tu, "Deeply-supervised nets," in *AISTATS*, 2015.

[47] J. Kim, J. K. Lee, and K. M. Lee, "Deeply-recursive convolutional network for image super-resolution," in *CVPR*, 2016.

[48] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *NIPS*, 2013, pp. 3111–3119.

[49] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, "mixup: Beyond empirical risk minimization," *arXiv preprint*, 2017.

[50] V. Verma, A. Lamb, C. Beckham, A. Najafi, I. Mitliagkas, A. Courville, D. Lopez-Paz, and Y. Bengio, "Manifold mixup: Better representations by interpolating hidden states," *arXiv preprint arXiv:1806.05236*, 2018.